DOCUMENT RESUME

ED 405 813 IR 018 253

AUTHOR Connolly, Mary V.
TITLE Starting Computer Science Using C++ with Objects: A Workable Approach.
PUB DATE 96
NOTE 5p.; In: Association of Small Computer Users in Education (ASCUE) Summer Conference Proceedings (29th, North Myrtle Beach, SC, June 9-13, 1996); see IR 018 247.
PUB TYPE Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Science Education; Computer Software Selection; *Curriculum Development; Educational Change; Higher Education; *Introductory Courses; Programming; *Programming Languages; Student Attitudes; Textbook Selection
IDENTIFIERS *C Programming Language; Object Oriented Programming; *Saint Marys College IN

ABSTRACT

Saint Mary's College (Indiana) offers a minor program in computer science. The program's introductory computer science class traditionally taught Pascal. The decision to change the introductory programming language to C++ with an object oriented approach was made when it became clear that there were good texts available for beginning students. Many students do not begin their study of computer science with a strong ability to handle symbolic languages and abstraction. Students moving to the data structures course, which is taught using an object oriented paradigm, have found the paradigm shift difficult even without a change of programming language. Accomplishments and problems were experienced during the first two offerings of the revised class. The students enjoyed using C++ and appreciated the experience of a real world programming environment. From the student point of view, object oriented programming is just as "natural" as function oriented programming, and it was not difficult to implement the basic themes of the introductory computer science course using C++. Using C++ with objects in the introductory computer science course is effective, provides an experience needed in the market place, and has positive benefits for the rest of the computer science curriculum. The paper also describes the selection of an introductory textbook and compiler program. (Contains 13 references.) (Author/SWC)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

Starting Computer Science Using C++ with Objects: A Workable Approach

Mary V. Connolly
Mathematics/Computer Science Department
Saint Mary's College
Notre Dame, Indiana
connolly@saintmarys.edu

ED 405 813

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
C.P. Singer
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

The curriculum for Computer Science I was well defined in Computing Curricula 1991 [1], but the language (and paradigm) issue continues to be debated extensively. Pascal, a favorite language with many instructors, was designed to help students learn to program in a structured, disciplined way, but has never found wide use outside of academia. Still, during a panel discussion on first languages at the SIGCSE Technical Symposium in 1993, an informal survey showed the vast majority of the approximately 200 people in the room used Pascal. Speakers presented eloquent reasons for change, but instructors continued to use Pascal. In contrast, a post symposium workshop at SIGCSE in 1995 entitled "Learning and Teaching C++ for the Pascal Generation" drew 106 participants[2]. Dick Reid of Michigan State University maintains a list of languages used in the first course taken by the majority of computer science students at participating colleges and universities. In results reported in January, 1996, out of 442 schools reporting, 35.5% use Pascal, 16.5% use Ada, 11.3% use Scheme, 8.8% use C, 7.9% use Modula, 7.7% use C++ and 2.9% use Modula-2. Sixteen other languages were included, each with a percentage less than two. It is important to realize that schools chose to participate in this survey which was done via the Internet.

At Saint Mary's College, any decision on the first language has to take into account the kind of student who enrolls in CS I. Computer Science at Saint Mary's is offered as a minor program; students in the program major in a variety of fields from mathematics and business to history, communications and philosophy. Many of these students ultimately are employed in the computer field, but do not begin their study of computer science with a strong ability to handle symbolic languages and abstraction. Also, since the college offers a minor program, there is not enough time to teach a variety of languages. Students moving to the data structures course, a course which is taught using an object oriented paradigm, have found the paradigm shift difficult even without a change of language. CS I at Saint Mary's is taught with a two hour closed lab, supported by two fifty minute lectures. Students learn by actively writing and testing programs; hence a good supportive text is essential.

At Saint Mary's the decision to change to C++ with an object oriented approach was made when it became clear that good texts were available, texts which were written for beginning students, which followed the Computing Curricula 1991 guidelines, and which used an object oriented approach. Although there are many books available which introduce an experienced programmer to C++, this course demands one in which design principles, programming syntax, program efficiency and the other major ideas of CS I are introduced at a level appropriate for the beginning computer science student.

IR018353

BEST COPY AVAILABLE

There is now a real choice of textbooks using C++ which are appropriate for a beginning course. The differences are usually apparent when one considers whether objects are introduced early in the book or later. A good example of the early approach is the book written by Decker and Hirshfield [3]. Classes are emphasized from the beginning. In contrast, the book by Adams, Leestma and Nyhoff introduces objects much later, almost requiring students to make a paradigm shift[4]. Since one of the goals at Saint Mary's was to avoid forcing students to make a paradigm shift in a later course, the text chosen, Computing Fundamentals with C++ by Rick Mercer, was one in which object language is used from the beginning[5]. Students use the correct words from the start, even though it is a few weeks into the course before the design implications are clear. After all the debate on the language itself, it came as a very pleasant surprise that C++ was no more difficult for students than Pascal. Beginning students do not know one language from another; nothing seems "natural" at the beginning. Perhaps all the language anxiety, if it exists, rests with instructors who are comfortable with a particular language. There are now several additional texts available, giving an individual instructor a good deal of choice [6,7,8,9,10,11,12, 13].

Another big decision involved the choice of a compiler. Saint Mary's opted to use Turbo C++ 3.1 for Windows. The campus network was already set up to handle Windows applications at the time, and many students came to the course comfortable with the Windows environment. Also, this choice does introduce students to a fairly realistic real world environment. One unexpected problem was that realistic real world environment. When programs did not run due to the usual variety of student errors, correcting and recompiling the code did not always result in a running program. The Windows environment did not always reset memory to allow the newly corrected program to run. This was frustrating, but it did prepare students for the less than friendly programming environment they might encounter on the job.

The course at Saint Mary's begins using built in classes (much as the old Pascal course used built in types) to develop elementary programs using selection and repetitive statements. Students are introduced to function prototypes by the second week; the old Pascal course had used an early introduction to procedures. One difference is that the object terminology is used from the start, even though students do not understand the concept of a class at this stage. After an introduction to a simple array, students see the need to develop classes and they begin to develop their own classes. For at least the last half of the course class time is spent designing classes and lab time is spent implementing and using them. For example, one lab has students explore the efficiency of the linear search versus the binary search by implementing a class for an array of integers which has both searches and a selection sort as methods. Gradually students begin to understand what a class really is; they realize that the design phase of a programming assignment requires an early identification of the needed classes. By the time two dimensional arrays are introduced, students automatically think about an appropriate class. It should be noted that none of the projects in the course makes use of inheritance; that is left to the data structures course.

One of the more difficult problems in the course involved how to handle character strings. Even in the old Pascal course, this was handled relatively late in the course since the string implementation is non standard in Pascal, usually quite dependent on the editing/compiling software being used. However, it is annoying to assign programs which involve bank records, inventory control or sales processing and never name people or things. Most texts get around the problem by author supplied string classes, again non standard. Although this is fine for simple programs, it is

more problematic when students are developing their own classes. Author supplied files do not always attach to student developed classes correctly. The first time the course was taught at Saint Mary's, the students were well into class development when the problems surfaced; the decision was made to wait until pointers were introduced so that students could handle character strings in their own classes using char * objects and the functions available in string.h. During the second offering of the course, all consideration of character strings was delayed until pointers were introduced. In retrospect this seems better than handing students code which works but is non standard and hides an understanding of how the machine handles character strings.

One unanticipated problem the first time the course was taught was the length of the labs. Although lab exercises were designed to be completed in the two hour period, students frequently needed three or more hours. Part of this was surely due to the instructor's inexperience with student errors in C++, since this was the first use of the language. Another part of the problem was the Windows environment discussed above. Both the instructor and a lab assistant (a student literally getting trained on the job) were present to assist students. As a group the students understood the difficulties involved in such a major change in a course and were quite patient. Although the situation improved during the second offering, labs still tended to be on the long side.

One delightful surprise was how much the students liked what they were doing. Many were positively energized by their work. They appreciated the fact that this was a real world programming environment and were willing to put up with the long labs. At first, it was not clear how the CS I model curriculum would work with the C++ with objects approach, but in fact it is not at all difficult to implement the basic themes using C++. From the student point of view, object oriented programming is just as "natural" as function oriented programming. It is important to observe that no single course will ever introduce a student to all the capabilities of the C++ language. Individual instructors must make decisions about what belongs in a "sane" subset of C++, a subset which gives the students enough of the language so they are able to complete reasonable projects without being overwhelmed by language details.

A shift to C++ with objects in the first course clearly has implications for the rest of the curriculum. Since the object oriented paradigm is not used in all Saint Mary's computer courses, students may have some adjustments to make in later courses. The Assembly Language course at Saint Mary's incorporates C; obviously students entering this course will already know some rudiments of the language whereas before they had to begin with an intense introduction to the language. The data structures course has used an object oriented approach for several years, but it has been implemented in Pascal with units. It will be changed so that the language used is C++, making a much better transition, particularly since a paradigm shift in design will no longer be necessary. In general, the curriculum should be more unified than before.

It clearly was time to change, and the new course is even more exciting than the first Pascal course was years ago. C++ with objects in the first course works, it provides an experience which is needed in the market place and has positive benefits for the rest of the curriculum.

Bibliography:

[1] ACM/IEEE-CS Joint Curriculum Task Force, Tucker, Allen B. (editor). Computing Curricula 1991. Association for Computing Machinery, 1991.

[2] Decker, Rick and Hirshfield, Stuart H. Learning and Teaching C++ for the Pascal Generation. SIGCSE Technical Symposium, March, 1995.

[3] Decker, R. and Hirshfield, S. The Object Concept. PWS Publishing, Boston, Massachusetts, 1995.

[4] Adams, J., Leestma, S. and Nyhoff, L. Turbo C++: An Introduction to Computing. Prentice-Hall, Upper Saddle River, New Jersey, 1996.

[5] Mercer, R. Computing Fundamentals with C++: Using, Modifying and Implementing Object Classes. Franklin, Beedle & Associates, Inc., Wilsonville, Oregon, 1995.

[6] Astrachan, O. L. A Computer Science Tapestry: Exploring Programming and Computer Science with C++. McGraw-Hill, 1996.

[7] Dale, N., Weems, C. and Headington, M. Programming and Problem Solving with C++. D. C. Heath & Company, Lexington, Massachusetts, 1996.

[8] Deitel, H. M. and Deitel, P. J. C++ How to Program. Prentice-Hall, Upper Saddle River, New Jersey, 1994.

[9] Friedman, F. L. and Koffman, E. B. Problem Solving, Abstraction, and Design using C++. Addison-Wesley, 1994.

[10] Lambert, K. A., Nance, D. W. and Naps, T. L. Introduction to Computer Science with C++. West Publishing Company, St. Paul, Minnesota, 1996.

[11] Levin, H. D. and Perry, J. E. An Introduction to Object-Oriented Design in C++. Addison-Wesley, 1996.

[12] Staugaard, A. C. Jr. Structuring Techniques: An Introduction using Turbo C++. Prentice-Hall, Upper Saddle River, New Jersey, 1995.

[13] Tucker, Allen B., et al. Fundamentals of Computing I: C++ Ed. McGraw-Hill, 1995.